```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIIII           VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIIII           VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIIII           VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

```
XX      XX    AAAAAA    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX    AAAAAA    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      AA      AA  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      AA      AA  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    AAAAAAAAAA  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    AAAAAAAAAA  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  AA      AA  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  AA      AA  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  AA      AA  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
XX      XX  AA      AA  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(2)	81	External and local symbol definitions
(3)	225	Device Driver Tables
(4)	285	XA_CONTROL_INIT, Controller initialization
(5)	332	XA_READ_WRITE, FDT for device data transfers
(6)	410	XA_SETMODE, Set Mode, Set characteristics FDT
(7)	477	XA_START, Start I/O routines
(10)	931	DR11-W DEVICE TIME-OUT
(11)	974	XA_INTERRUPT, Interrupt service routine for DR11-W
(12)	1047	XA_REGISTER - Handle DR11-W CSR transfers
(13)	1092	XA_CANCEL, Cancel I/O routine
(14)	1147	DEL_ATTNAST, Deliver ATTN AST's
(15)	1200	XA_REGDUMP - DR11-W register dump routine
(16)	1256	XA_DEV_RESET - Device reset DR11-W

```
0000      .TITLE  XADRIVER - VAX/VMS DR11-W DRIVER
0000      .IDENT  'V04-001'
0000
0000  ;
0000  ;*****************************************************************************
0000  ;*                                                                           *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000  ;*  ALL RIGHTS RESERVED.                                                     *
0000  ;*                                                                           *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000  ;*  TRANSFERRED.                                                             *
0000  ;*                                                                           *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000  ;*  CORPORATION.                                                             *
0000  ;*                                                                           *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000  ;*                                                                           *
0000  ;*                                                                           *
0000  ;*****************************************************************************
0000  ;
0000  ;++
0000  ;
0000  ; FACILITY:
0000  ;
0000  ;       VAX/VMS Executive, I/O Drivers
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       This module contains the DR11-W driver:
0000  ;
0000  ;               Tables for loading and dispatching
0000  ;               Controller initialization routine
0000  ;               FDT routine
0000  ;               The start I/O routine
0000  ;               The interrupt service routine
0000  ;               Device specific Cancel I/O
0000  ;               Error logging register dump routine
0000  ;
0000  ; ENVIRONMENT:
0000  ;
0000  ;       Kernal Mode, Non-paged
0000  ;
0000  ; AUTHOR:
0000  ;
0000  ;       C. A. Sameulson 10-JAN-79
0000  ;
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V04-001 JLV0395         Jake VanNoy             6-SEP-1984
```

```
0000    58 ;                Add AVL bit to DEVCHAR.
0000    59 ;
0000    60 ;        V03-006 TMK0001         Todd M. Katz            07-Dec-1983
0000    61 ;                Fix a broken branch.
0000    62 ;
0000    63 ;        V03-005 JLV0304         Jake VanNoy             24-AUG-1983
0000    64 ;                Several bug fixes. All word writes to XA_CSR now have
0000    65 ;                ATTN set so as to prevent lost interrupts. Attention
0000    66 ;                AST list is synchronized at device IPL in DEL_ATTNAST.
0000    67 ;                Correct status is returned on a set mode ast that
0000    68 ;                is returns through EXE$FINISHIO. REQCOM's are always
0000    69 ;                done at FIPL.  Signed division that prevented full size
0000    70 ;                transfers has been fixed.
0000    71 ;
0000    72 ;
0000    73 ;        V03-004 KDM0059         Kathleen D. Morse       14-Jul-1983
0000    74 ;                Change time-wait loops to use new TIMEDWAIT macro.
0000    75 ;                Add $DEVDEF.
0000    76 ;
0000    77 ;        V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    78 ;                Added $DYNDEF, $DCDEF, and $SSDEF.
0000    79 ;--
```

```
                 0000          .SBTTL  External and local symbol definitions

                 0000  ; External symbols

                 0000          $ACBDEF                         ; AST control block
                 0000          $CRBDEF                         ; Channel request block
                 0000          $DCDEF                          ; Device types
                 0000          $DDBDEF                         ; Device data block
                 0000          $DEVDEF                         ; Device characteristics
                 0000          $DPTDEF                         ; Driver prolog table
                 0000          $DYNDEF                         ; Dynamic data structure types
                 0000          $EMBDEF                         ; EMB offsets
                 0000          $IDBDEF                         ; Interrupt data block
                 0000          $IODEF                          ; I/O function codes
                 0000          $IPLDEF                         ; Hardware IPL definitions
                 0000          $IRPDEF                         ; I/O request packet
                 0000          $PRDEF                          ; Internal processor registers
                 0000          $PRIDEF                         ; Scheduler priority increments
                 0000          $SSDEF                          ; System status codes
                 0000          $UCBDEF                         ; Unit control block
                 0000          $VECDEF                         ; Interrupt vector block
                 0000          $XADEF                          ; Define device specific characteristics

                 0000  ; Local symbols

                 0000  ; Argument list (AP) offsets for device-dependent QIO parameters

00000000         0000  P1      = 0                             ; First QIO parameter
00000004         0000  P2      = 4                             ; Second QIO parameter
00000008         0000  P3      = 8                             ; Third QIO parameter
0000000C         0000  P4      = 12                            ; Fourth QIO parameter
00000010         0000  P5      = 16                            ; Fifth QIO parameter
00000014         0000  P6      = 20                            ; Sixth QIO parameter

                 0000  ; Other constants

0000000A         0000  XA_DEF_TIMEOUT  = 10                    ; 10 second default device timeout
0000FFFF         0000  XA_DEF_BUFSIZ   = 65535                 ; Default buffer size
00000001         0000  XA_RESET_DELAY  = <<2+9>/10>            ; Delay N microseconds after RESET
                 0000                                          ;  (rounded up to 10 microsec intervals)

                 0000  ; DR11-W definitions that follow the standard UCB fields
                 0000  ; *** N O T E ***  ORDER OF THESE UCB FIELDS IS ASSUMED

                 0000          $DEFINI UCB
000000A0         0000          .=UCB$L_DPC+4
                 00A0  $DEF    UCB$L_XA_ATTN                   ; Attention AST listhead
000000A4         00A0                  .BLKL   1
                 00A4  $DEF    UCB$W_XA_CSRTMP                 ; Temporary storage of CSR image
000000A6         00A4                  .BLKW   1
                 00A6  $DEF    UCB$W_XA_BARTMP                 ; Temporary storage of BAR image
000000A8         00A6                  .BLKW   1
                 00A8  $DEF    UCB$W_XA_CSR                    ; Saved CSR on interrupt
000000AA         00A8                  .BLKW   1
                 00AA  $DEF    UCB$W_XA_EIR                    ; Saved EIR on interrupt
000000AC         00AA                  .BLKW   1
```

```
                00AC   $DEF    UCB$W_XA_IDR                    ; Saved IDR on interrupt
000000AE        00AC                   .BLKW   1
                00AE   $DEF    UCB$W_XA_BAR                    ; Saved BAR register on interrupt
000000B0        00AE                   .BLKW   1
                00B0   $DEF    UCB$W_XA_WCR                    ; Saved WCR register on interrupt
000000B2        00B0                   .BLKW   1
                00B2   $DEF    UCB$W_XA_ERROR                  ; Saved device status flag
000000B4        00B2                   .BLKW   1
                00B4   $DEF    UCB$L_XA_DPR                    ; Data Path Register contents
000000B8        00B4                   .BLKL   1
                00B8   $DEF    UCB$L_XA_FMPR                   ; Final Map Register contents
000000BC        00B8                   .BLKL   1
                00BC   $DEF    UCB$L_XA_PMPR                   ; Previous Map Register contents
000000C0        00BC                   .BLKL   1
                00C0   $DEF    UCB$W_XA_DPRN                   ; Saved Datapath Register Number
000000C2        00C0                   .BLKW   1               ; And Datapath Parity error flag
                00C2
                00C2   ; Bit positions for device-dependent status field in UCB
                00C2
                00C2           $VIELD  UCB,0,<-                ; UCB device specific bit definitions
                00C2                   <ATTNAST,,M>,-          ; ATTN AST requested
                00C2                   <UNEXPT,,M>,-           ; Unexpected interrupt received
                00C2                   >
000000C2        00C2   UCB$K_SIZE=.
                00C2           $DEFEND UCB
                0000
                0000   ; Device register offsets from CSR address
                0000
                0000           $DEFINI XA                      ; Start of DR11-W definitions
                0000   $DEF    XA_WCR                          ; Word count
00000002        0000                   .BLKW   1
                0002   $DEF    XA_BAR                          ; Buffer address
00000004        0002                   .BLKW   1
                0004   $DEF    XA_CSR                          ; Control/status
                0004
                0004
                0004   ; Bit positions for device control/status register
                0004
                0004           $EQULST XA$K_,,0,1,<-           ; Define CSR FNCT bit values
                0004                   <FNCT1,2>-
                0004                   <FNCT2,4>-
                0004                   <FNCT3,8>-
                0004                   <STATUSA,2048>-         ; Define CSR STATUS bit values
                0004                   <STATUSB,1024>-
                0004                   <STATUSC,512>-
                0004           >
                0004
                0004           $VIELD  XA_CSR,0,<-             ; Control/status register
                0004                   <GO,,M>,-               ; Start device
                0004                   <FNCT,3,M>,-            ; CSR FNCT bits
                0004                   <XBA,2,M>,-             ; Extended address bits
                0004                   <IE,,M>,-               ; Enable interrupts
                0004                   <RDY,,M>,-              ; Device ready for command
                0004                   <CYCLE,,M>,-            ; Starts slave transmit
                0004                   <STATUS,3,M>,-          ; CSR STATUS bits
                0004                   <MAINT,,M>,-            ; Maintenance bit
                0004                   <ATTN,,M>,-             ; Status from other processor
```

```
                0004   195                     <NEX,,M>,-                  ; Nonexistent memory flag
                0004   196                     <ERROR,,M>,-                ; Error or external interrupt
                0004   197             >
                0004   198 $DEF        XA_EIR                              ; Error information register
                0004   199
                0004   200 ; Bit positions for error information register
                0004   201
                0004   202             $VIELD  XA_EIR,0,<-                 ; Error information register
                0004   203                     <REGFLG,,M>,-               ; Flags whether EIR or CSR is accessed
                0004   204                     <SPARE,7,M>,-               ; Unused - spare
                0004   205                     <BURST,,M>,-                ; Burst mode transfer occured
                0004   206                     <DLT,,M>,-                  ; Time-out for successive burst xfer
                0004   207                     <PAR,,M>,-                  ; Parity error during DATI/P
                0004   208                     <ACLO,,M>,-                 ; Power fail on this processor
                0004   209                     <MULTI,,M>,-                ; Multi-cycle request error
                0004   210                     <ATTN,,M>,-                 ; ATTN - same as in CSR
                0004   211                     <NEX,,M>,-                  ; NEX - same as in CSR
                0004   212                     <ERROR,,M>,-                ; ERROR - same as in CSR
                0004   213             >
00000006        0004   214                     .BLKW   1
                0006   215
                0006   216 $DEF        XA_IDR                              ; Input Data Buffer register
                0006   217 $DEF        XA_ODR                              ; Output Data Buffer register
00000008        0006   218                     .BLKW   1
                0008   219
                0008   220             $DEFEND XA                          ; End of DR11-W definitions
                0000   221
                0000   222
                0000   223
```

```
0000          .SBTTL  Device Driver Tables

; Driver prologue table

        DPTAB   -                                       ; DPT-creation macro
                END=XA_END,-                            ; End of driver label
                ADAPTER=UBA,-                           ; Adapter type
                FLAGS=DPT$M_SVP,-                       ; Allocate system page table
                UCBSIZE=UCB$K_SIZE,-                    ; UCB size
                NAME=XADRIVER                           ; Driver name
0038    DPT_STORE INIT                                  ; Start of load
                                                        ; initialization table
        DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Device fork IPL
003C    DPT_STORE UCB,UCB$B_DIPL,B,22                   ; Device interrupt IPL
0040    DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
                DEV$M_AVL!-                             ; Available
                DEV$M_RTM!-                             ; Real Time device
                DEV$M_ELG!-                             ; Error Logging enabled
                DEV$M_IDV!-                             ;   input device
                DEV$M_ODV>                              ;   output device
0047    DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; Device class
004B    DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DR11W         ; Device Type
004F    DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,-               ; Default buffer size
                XA_DEF_BUFSIZ
0054    DPT_STORE REINIT                                ; Start of reload
                                                        ; initialization table
        DPT_STORE DDB,DDB$L_DDT,D,XA$DDT                ; Address of DDT
0059    DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
                XA_INTERRUPT                            ; service routine
005E    DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,-        ; Address of controller
                D,XA_CONTROL_INIT                       ; initialization routine
0063    DPT_STORE END                                   ; End of initialization
                                                        ; tables

; Driver dispatch table

        DDTAB   -                                       ; DDT-creation macro
                DEVNAM=XA,-                             ; Name of device
                START=XA_START,-                        ; Start I/O routine
                FUNCTB=XA_FUNCTABLE,-                   ; FDT address
                CANCEL=XA_CANCEL,-                      ; Cancel I/O routine
                REGDMP=XA_REGDUMP,-                     ; Register dump routine
                DIAGBF=<<T3*4>+<<3+5+1>*4>>,-           ; Diagnostic buffer size
                ERLGBF=<<13*4>+<1*4>+<EMB$L_DV_REGSAV>> ; Error log buffer size

;
; Function dispatch table
;
XA_FUNCTABLE:                                           ; FDT for driver
        FUNCTAB ,-                                      ; Valid I/O functions
                <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK,-
                SETMODE,SETCHAR,SENSEMODE,SENSECHAR>
        FUNCTAB ,                                       ; No buffered functions
        FUNCTAB XA_READ_WRITE,-                         ; Device-specific FDT
                <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK>
        FUNCTAB +EXE$READ,<READPBLK,READLBLK,READVBLK>
        FUNCTAB +EXE$WRITE,<WRITEPBLK,WRITELBLK,WRITEVBLK>
```

```
006C   282        FUNCTAB XA_SETMODE,<SETMODE,SETCHAR>
0078   283        FUNCTAB +EXE$SENSEMODE,<SENSEMODE,SENSECHAR>
```

```
                        0084          .SBTTL  XA_CONTROL_INIT, Controller initialization
                        0084
                        0084  ;++
                        0084  ; XA_CONTROL_INIT, Called when driver is loaded, system is booted, or
                        0084  ; power failure recovery.
                        0084  ;
                        0084  ; Functional Description:
                        0084  ;
                        0084  ;       1) Allocates the direct data path permanently
                        0084  ;       2) Assigns the controller data channel permanently
                        0084  ;       3) Clears the Control and Status Register
                        0084  ;       4) If power recovery, requests device time-out
                        0084  ;
                        0084  ; Inputs:
                        0084  ;
                        0084  ;       R4 = address of CSR
                        0084  ;       R5 = address of IDB
                        0084  ;       R6 = address of DDB
                        0084  ;       R8 = address of CRB
                        0084  ;
                        0084  ; Outputs:
                        0084  ;
                        0084  ;       VEC$V_PATHLOCK bit set in CRB$L_INTD+VEC$B_DATAPATH
                        0084  ;       UCB address placed into IDB$L_OWNER
                        0084  ;
                        0084  ;
                        0084  ;--
                        0084
                        0084  XA_CONTROL_INIT:
                        0084
                        0084          MOVL    IDB$L_UCBLST(R5),R0     ; Address of UCB
   50    18 A5    D0    0088          MOVL    R0,IDB$L_OWNER(R5)      ; Make permanent controller owner
   04 A5    50    D0    008C          BISW    #UCB$M_ONLINE,UCB$W_STS(R0)
                        0090                                          ; Set device status "on-line"
                        0090
                        0090  ; If powerfail has occured and device was active, force device time-out.
                        0090  ; The user can set his own time-out interval for each request.  Time-
                        0090  ; out is forced so a very long time-out period will be short circuited.
                        0090
05 64 A0    05    E0    0090          BBS     #UCB$V_POWER,UCB$W_STS(R0),10$
                        0095                                          ; Branch if powerfail
37 A8    80 8F    88    0095          BISB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R8)
                        009A                                          ; Permanently allocate direct datapath
                        009A  10$:
         0561     30    009A          BSBW    XA_DEV_RESET            ; Reset DR11W
                  05    009D          RSB                             ; Done
```

```
                                009E          .SBTTL  XA_READ_WRITE, FDT for device data transfers
                                009E
                                009E    ;++
                                009E    ; XA_READ_WRITE, FDT for READLBLK,READVBLK,READPBLK,WRITELBLK,WRITEVBLK,
                                009E    ;                         WRITEPBLK
                                009E    ;
                                009E    ; Functional description:
                                009E    ;
                                009E    ;       1) Rejects QUEUE I/O's with odd transfer count
                                009E    ;       2) Rejects QUEUE I/O's for BLOCK MODE request to UBA Direct Data
                                009E    ;          PATH on odd byte boundary
                                009E    ;       3) Stores request time-out count specified in P3 into IRP
                                009E    ;       4) Stores FNCT bits specified in P4 into IRP
                                009E    ;       5) Stores word to write into ODR from P5 into IRP
                                009E    ;       6) Checks block mode transfers for memory modify access
                                009E    ;
                                009E    ; Inputs:
                                009E    ;
                                009E    ;       R3 = Address of IRP
                                009E    ;       R4 = Address of PCB
                                009E    ;       R5 = Address of UCB
                                009E    ;       R6 = Address of CCB
                                009E    ;       R8 = Address of FDT routine
                                009E    ;       AP = Address of P1
                                009E    ;               P1 = Buffer Address
                                009E    ;               P2 = Buffer size in bytes
                                009E    ;               P3 = Request time-out period (conditional on IO$M_TIMED)
                                009E    ;               P4 = Value for CSR FNCT bits (conditional on IO$M_SETFNCT)
                                009E    ;               P5 = Value for ODR (conditional on IO$M_SETFNCT)
                                009E    ;               P6 = Address of Diagnostic Buffer
                                009E    ;
                                009E    ; Outputs:
                                009E    ;
                                009E    ;       R0 = Error status if odd transfer count
                                009E    ;       IRP$L_MEDIA = Time-out count for this request
                                009E    ;       IRP$L_SEGVBN = FNCT bits for DR11-W CSR and ODR image
                                009E    ;
                                009E    ;--
                                009E
                                009E    XA_READ_WRITE:
          09 04 AC    E9        009E            BLBC    P2(AP),10$              ; Branch if transfer count even
             50    14   3C      00A2    2$:     MOVZWL  #SS$_BADPARAM,R0        ; Set error status code
     00000000'GF    17          00A5    5$:     JMP     G^EXE$ABORTIO           ; Abort request
          51    20 A3   3C      00AB    10$:    MOVZWL  IRP$W_FUNC(R3),R1       ; Fetch I/O Function code
       38 A3    08 AC   D0      00AF            MOVL    P3(AP),IRP$L_MEDIA(R3)  ; Set request specific time-out count
          04 51    07   E0      00B4            BBS     #IO$V_TIMED,R1,15$      ; Branch if time-out specified
          38 A3    0A   D0      00B8            MOVL    #XA_DEF_TIMEOUT,IRP$L_MEDIA(R3)
                                00BC                                            ; Else set default timeout value
          14 51    08   E1      00BC    15$:    BBC     #IO$V_DIAGNOSTIC,R1,20$ ; Branch if not maintenance reqeust
51        51    06   00   EF    00C0            EXTZV   #IO$V_FCODE,#IO$S_FCODE,R1,R1 ; AND out all function modifiers
             51    0C   91      00C5            CMPB    #IO$_READPBLK,R1        ; If maintenance function, must be
                                00C8                                            ; physical I/O read or write
                   0A   13      00C8            BEQL    20$
             51    0B   91      00CA            CMPB    #IO$_WRITEPBLK,R1
                   05   13      00CD            BEQL    20$
             50    24   3C      00CF            MOVZWL  #SS$_NOPRIV,R0          ; No privilege for operation
                   D1   11      00D2            BRB     5$                      ; Abort request
```

```
50   0C AC   03   00   EF  00D4   389 20$:    EXTZV   #0,#3,P4(AP),R0          ; Get value for FNCT bits
     48 A3   50   01   78  00DA   390          ASHL    #XA_CSR$V_FNCT,R0,IRP$L_SEGVBN(R3) ; Shift into position for CSR
       4A A3   10 AC   B0  00DF   391          MOVW    P5(AP),IRP$L_SEGVBN+2(R3) ; Store ODR value for later
                                  00E4   392
                                  00E4   393 ; If this is a block mode transfer, check buffer for modify access
                                  00E4   394 ; whether or not the function is read or write.  The DR11-W does
                                  00E4   395 ; not decide whether to read or write, the users device does.
                                  00E4   396 ; For word mode requests, return to read check or write check.
                                  00E4   397 ;
                                  00E4   398 ; If this is a BLOCK MODE request and the UBA Direct Data Path is
                                  00E4   399 ; in use, check the data buffer address for word alignment.  If buffer
                                  00E4   400 ; is not word aligned, reject the request.
                                  00E4   401
       0E 20 A3   06   E0  00E4   402          BBS     #IO$V_WORD,IRP$W_FUNC(R3),30$
                                  00E9   403                                  ; Branch if word mode transfer
       03 44 A5   00   E0  00E9   404          BBS     #XA$V_DATAPATH,UCB$L_DEVDEPEND(R5),25$
                                  00EE   405                                  ; Branch if Buffered Data Path in use
              B1 6C   E8  00EE   406          BLBS    P1(AP),2$               ; DDP, branch on bad alignment
         00000000'GF   17  00F1   407 25$:    JMP     G^EXE$MODIFY            ; Checke buffer for modify access
                       05  00F7   408 30$:    RSB                             ; Return
```

```
                        00F8          .SBTTL  XA_SETMODE, Set Mode, Set characteristics FDT
                        00F8
                        00F8  ;++
                        00F8  ; XA_SETMODE, FDT routine to process SET MODE and SET CHARACTERISTICS
                        00F8  ;
                        00F8  ; Functional description:
                        00F8  ;
                        00F8  ;       If IO$M_ATTNAST modifier is set, queue attention AST for device
                        00F8  ;       If IO$M_DATAPATH modifier is set, queue packet.
                        00F8  ;       Else, finish I/O.
                        00F8  ;
                        00F8  ; Inputs:
                        00F8  ;
                        00F8  ;       R3 = I/O packet address
                        00F8  ;       R4 = PCB address
                        00F8  ;       R5 = UCB address
                        00F8  ;       R6 = CCB address
                        00F8  ;       R7 = Function code
                        00F8  ;       AP = QIO Paramater list address
                        00F8  ;
                        00F8  ; Outputs:
                        00F8  ;
                        00F8  ;       If IO$M_ATTNAST is specified, queue AST on UCB attention AST list.
                        00F8  ;       If IO$M_DATAPATH is specified, queue packet to driver.
                        00F8  ;       Else, use exec routine to update device characteristics
                        00F8  ;
                        00F8  ;--
                        00F8
                        00F8  XA_SETMODE:
50    20 A3     3C      00F8          MOVZWL  IRP$W_FUNC(R3),R0       ; Get entire function code
2B 50    08     E1      00FC          BBC     #IO$V_ATTNAST,R0,20$    ; Branch if not an ATTN AST
                        0100
                        0100  ; Attention AST request
                        0100
     0090 8F     BB     0100          PUSHR   #^M<R4,R7>
57   00A0 C5     9E     0104          MOVAB   UCB$L_XA_ATTN(R5),R7    ; Address of ATTN AST control block list
 00000000'GF     16     0109          JSB     G^COM$SETATTNAST        ; Set up attention AST
     0090 8F     BA     010F          POPR    #^M<R4,R7>
        2D 50    E9     0113          BLBC    R0,50$                  ; Branch if error
   68 A5   01    A8     0116          BISW    #UCB$M_ATTNAST,UCB$W_DEVSTS(R5)
                        011A                                          ; Flag ATTN AST expected.
03 68 A5   01    E1     011A          BBC     #UCB$V_UNEXPT,UCB$W_DEVSTS(R5),10$
                        011F                                          ; Deliver AST if unsolicited interrupt
        045E     30     011F          BSBW    DEL_ATTNAST
     50   01     9A     0122  10$:    MOVZBL  #SS$_NORMAL,R0          ; Set status
 00000000'GF     17     0125          JMP     G^EXE$FINISHIOC         ; Thats all for now (clears R1)
                        012B
                        012B  ; If modifier IO$M_DATAPATH is set,
                        012B  ; queue packet.  The data path is changed at driver level to preserve
                        012B  ; order with other requests.
                        012B
   06 50    0A     E0     012B  20$:    BBS     S^#IO$V_DATAPATH,R0,30$ ; If BDP modifier set, queue packet
                        012F
 00000000'GF     17     012F          JMP     G^EXE$SETCHAR           ; Set device characteristics
                        0135
                        0135  ; This is a request to change data path useage, queue packet
                        0135
```

```
    57  1A  D1  0135   467 30$:    CMPL    #IO$_SETCHAR,R7          ; Set characteristics?
        06  12  0138   468         BNEQ    45$                      ; No, must have the privelege
00000000'GF  17  013A   469         JMP     G^EXE$SETMODE            ; Queue packet to start I/O
                0140   470
                0140   471 ; Error, abort IO
                0140   472
    50  24  3C  0140   473 45$:    MOVZWL  #SS$_NOPRIV,R0           ; No priv for operation
        51  D4  0143   474 50$:    CLRL    R1
00000000'GF  17  0145   475         JMP     G^EXE$ABORTIO            ; Abort IO on error
```

```
                        014B          .SBTTL  XA_START, Start I/O routines
                        014B  ;++
                        014B  ; XA_START - Start a data transfer, set characteristics, enable ATTN AST.
                        014B  ;
                        014B  ; Functional Description:
                        014B  ;
                        014B  ;       This routine has two major functions:
                        014B  ;
                        014B  ;       1) Start an I/O transfer.  This transfer can be in either word
                        014B  ;          or block mode.  The FNCTN bits in the DR11-W CSR are set.  If
                        014B  ;          the transfer count is zero, the STATUS bits in the DR11-W CSR
                        014B  ;          are read and the request completed.
                        014B  ;       2) Set Characteristics.  If the function is change data path, the
                        014B  ;          new data path flag is set in the UCB.
                        014B  ;
                        014B  ; Inputs:
                        014B  ;
                        014B  ;       R3 = Address of the I/O request packet
                        014B  ;       R5 = Address of the UCB
                        014B  ;
                        014B  ; Outputs:
                        014B  ;
                        014B  ;       R0 = final status and number of bytes transferred
                        014B  ;       R1 = value of CSR STATUS bits and value of input data buffer register
                        014B  ;       Device errors are logged
                        014B  ;       Diagnostic buffer is filled
                        014B  ;
                        014B  ;--
                        014B          .ENABL  LSB
                        014B
                        014B  XA_START:
                        014B
                        014B  ; Retrieve the address of the device CSR
                        014B
                        014B          ASSUME  IDB$L_CSR EQ 0
        54   24 A5   D0 014B          MOVL    UCB$L_CRB(R5),R4        ; Address of CRB
        54   2C B4   D0 014F          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                        0153                                          ; Address of CSR
                        0153
                        0153  ; Fetch the I/O function code
                        0153
        51   20 A3   3C 0153          MOVZWL  IRP$W_FUNC(R3),R1       ; Get entire function code
      009A C5   51   B0 0157          MOVW    R1,UCB$W_FUNC(R5)       ; Save FUNC in UCB for Error Logging
52    51   06   00   EF 015C          EXTZV   #IO$V_FCODE,#IO$S_FCODE,R1,R2 ; Extract function field
                        0161
                        0161  ; Dispatch on function code.  If this is SET CHARACTERISTICS, we will
                        0161  ; select a data path for future use.
                        0161  ; If this is a transfer function, it will either be processed in word
                        0161  ; or block mode.
                        0161
        52   1A   91    0161          CMPB    #IO$_SETCHAR,R2         ; Set characteristics?
             23   12    0164          BNEQ    3$
                        0166
                        0166  ;++
                        0166  ; SET CHARACTERISTICS - Process Set Characteristics QIO function
                        0166  ;
                        0166  ; INPUTS:
```

```
                        0166   ;
                        0166   ;       XA_DATAPATH bit in Device Characteristics specifies which data path
                        0166   ;       to use.  If bit is a one, use buffered data path.  If zero, use
                        0166   ;       direct datapath.
                        0166   ;
                        0166   ; OUTPUTS:
                        0166   ;
                        0166   ;       CRB is flagged as to which datapath to use.
                        0166   ;       DEVDEPEND bits in device characteristics is updated
                        0166   ;               XA_DATAPATH = 1 -> buffered data path in use
                        0166   ;               XA_DATAPATH = 0 -> direct data path in use
                        0166   ;--
                        0166
     50   24 A5   D0    0166           MOVL    UCB$L_CRB(R5),R0                ; Get CRB address
 40 A5   38 A3   7D    016A           MOVQ    IRP$L_MEDIA(R3),UCB$B_DEVCLASS(R5) ; Set device characteristics
 37 A0   80 8F   88    016F           BISB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0)
                       0174                                                   ; Assume direct datapath
 05 44 A5   00   E1    0174           BBC     #XA$V_DATAPATH,UCB$L_DEVDEPEND(R5),2$ ;  Were we right?
 37 A0   80 8F   8A    0179           BICB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0) ; Set buffered datapat
                       017E   2$:
         51      D4    017E           CLRL    R1                              ; Return Success
     50  01      3C    0180           MOVZWL  #SS$_NORMAL,R0
                       0183           REQCOM
                       0189
                       0189   ; If subfunction modifier for device reset is set, do one here
                       0189
     03 51   0B   E1   0189   3$:     BBC     S^#IO$V_RESET,R1,4$     ; Branch if not device reset
         046E    30    018D           BSBW    XA_DEV_RESET            ; Reset DR11-W
                       0190
                       0190   ; This must be a data transfer function - i.e. READ OR WRITE
                       0190   ; Check to see if this is a zero length transfer.
                       0190   ; If so, only set CSR FNCT bits and return STATUS from CSR
                       0190
        7E A5   B5     0190   4$:       TSTW    UCB$W_BCNT(R5)          ; Is transfer count zero?
           59   12     0193           BNEQ    10$                     ; No, continue with data transfer
     2E 51   09   E1   0195           BBC     S^#IO$V_SETFNCT,R1,6$   ; Set CSR FNCT specified?
                       0199           DSBINT
 06 A4   4A A3   B0    019F           MOVW    IRP$L_SEGVBN+2(R3),XA_ODR(R4)
                       01A4                                           ; Store word in ODR
     50  04 A4   3C    01A4           MOVZWL  XA_CSR(R4),R0
   50  800E 8F   AA    01A8           BICW    #<XA_CSR$M_FNCT!XA_CSR$M_ERROR>,R0
     50  48 A3   A8    01AD           BISW    IRP$L_SEGVBN(R3),R0
   50  2000 8F   A8    01B1           BISW    #XA_CSR$M_ATTN,R0       ; Force ATTN on to prevent lost interrupt
     04 A4   50   B0   01B6           MOVW    R0,XA_CSR(R4)
 05 44 A5   01   E1    01BA           BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),5$ ; Link mode?
04 A4   50   04   AB   01BF           BICW3   #XA$K_FNCT2,R0,XA_CSR(R4)       ; Make FNCT bit 2 a pulse
                       01C4   5$:
                       01C4           ENBINT
                       01C7   6$:
         032B    30    01C7           BSBW    XA_REGISTER             ; Fetch DR11-W registers
        06 50    E8    01CA           BLBS    R0,7$                   ; If error, then log it
  00000000'GF    16    01CD           JSB     G^ERL$DEVICERR          ; Log a device error
  00000000'GF    16    01D3   7$:     JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer if specified
 51   00A8 C5   D0    01D9           MOVL    UCB$W_XA_CSR(R5),R1     ; Return CSR and EIR in R1
 50   00B2 C5   3C    01DE           MOVZWL  UCB$W_XA_ERROR(R5),R0   ; Return status in R0
 04 A4   40 8F   88    01E3           BISB    #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts
                       01E8           REQCOM                          ; Request done
```

```
                        01EE
                        01EE  ; Build CSR image in R0 for later use in starting transfers
                        01EE
                        01EE  10$:
        50    7E A5   3C  01EE          MOVZWL  UCB$W_BCNT(R5),R0        ; Fetch byte count
00B4 C5   50    02    C7  01F2          DIVL3   #2,R0,UCB$L_XA_DPR(R5)   ; Make byte count into word count
                          01F8          ;
                          01F8          ; Set up UCB$W_CSRTMP used for loading CSR later
                          01F8          ;
        50    04 A4   3C  01F8          MOVZWL  XA_CSR(R4),R0
     50    FFF1 8F    AA  01FC          BICW    #^C<XA_CSR$M_FNCT>,R0
     50    2040 8F    A8  0201          BISW    #XA_CSR$M_IE!XA_CSR$M_ATTN,R0   ; Set Interrupt Enable and ATTN
        07 51    09   E1  0206          BBC     S^#IO$V_SETFNCT,R1,20$   ; Set FNCT bits in CSR?
           50    0E   AA  020A          BICW    #<XA_CSR$M_FNCT>,R0      ; Yes, Clear previous FNCT bits
        50    48 A3   88  020D          BISB    IRP$L_SEGVBN(R3),R0      ; OR in new value
        05 51    08   E1  0211  20$:    BBC     S^#IO$V_DIAGNOSTIC,R1,23$        ; Check for maintenance function
     50    1000 8F    A8  0215          BISW    #XA_CSR$M_MAINT,R0       ; Set maintenance bit in CSR image
                          021A
                          021A  ; Is this a word mode or block mode request?
                          021A
     00A4 C5    50    B0  021A  23$:    MOVW    R0,UCB$W_XA_CSRTMP(R5)   ; Save CSR image in UCB
        03 51    06   E1  021F          BBC     S^#IO$V_WORD,R1,BLOCK_MODE ; Check if word or block mode
              013A    31  0223          BRW     WORD_MODE                ; Branch to handle word mode
```

```
                                0226   ;++
                                0226   ; BLOCK MODE -- Process a Block Mode (DMA) transfer request
                                0226   ;
                                0226   ; FUNCTIONAL DESCRIPTION:
                                0226   ;
                                0226   ;       This routine takes the buffer address, buffer size, fucntion code,
                                0226   ;       and function modifier fields from the IRP.  It calculates the UNIBUS
                                0226   ;       address, allocates the UBA map registers, loads the DR11-W device
                                0226   ;       registers and starts the request.
                                0226   ;--
                                0226   ; Set up UBA
                                0226   ; Start transfer
                                0226
                                0226   BLOCK_MODE:
                                0226
                                0226   ; If IO$M_CYCLE subfunction is specified, set CYCLE bit in CSR image
                                0226
          07 51    0C    E1     0226           BBC     #IO$V_CYCLE,R1,25$       ; Set CYCLE bit in CSR?
   00A4 C5    0100 8F    A8     022A           BISW    #XA_CSR$M_CYCLE,UCB$W_XA_CSRTMP(R5) ; If yes, or into CSR image
                                0231
                                0231   ; Allocate UBA data path and map registers
                                0231
                                0231   25$:
                                0231           REQDPR                           ; Request UBA data path
                                0237           REQMPR                           ; Request UBA map registers
                                023D           LOADUBA                          ; Load UBA map registers
                                0243
                                0243   ; Calculate the UNIBUS transfer address for the DR11-W from the UBA
                                0243   ; map register address and byte offset.
                                0243
          51    7C A5    3C     0243           MOVZWL  UCB$W_BOFF(R5),R1        ; Byte offset in first page of xfer
          52    24 A5    D0     0247           MOVL    UCB$L_CRB(R5),R2         ; Address of CRB
 51   09    09    34 A2    F0   024B           INSV    CRB$L_INTD+VEC$W_MAPREG(R2),#9,#9,R1
                                0251                                            ; Insert page number
     52   51    02    10    EF   0251           EXTZV   #16,#2,R1,R2             ; Extract bits 17:16 of bus address
          52    52    04    78   0256           ASHL    #XA_CSR$V_XBA,R2,R2      ; Shift extended memeroy bits for CSR
                52    01    AB   025A           BISW    #XA_CSR$M_GO,R2          ; Set "GO" bit into CSR image
          00A4 C5    52    AB   025D           BISW    R2,UCB$W_XA_CSRTMP(R5)   ; Set into CSR image we are building
50   00A4 C5    0101 8F    AB   0262           BICW3   #<XA_CSR$M_GO!XA_CSR$M_CYCLE>,UCB$W_XA_CSRTMP(R5),R0
                                026A                                            ; CSR image less "GO" and "CYCLE"
     52   00A4 C5    04    AB   026A           BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),R2 ; CSR image less FNCT bit 2
          00A6 C5    51    B0   0270           MOVW    R1,UCB$W_XA_BARTMP(R5)   ; Save BAR for error logging
                                0275
                                0275   ; At this juncture:
                                0275   ;       R0 = CSR image less "GO" and "CYCLE"
                                0275   ;       R1 = low 16 bits of transfer bus address
                                0275   ;       R2 = CSR image less FNCT bit 2
                                0275   ;       UCB$L_XA_DPR(R5) = transfer count in words
                                0275   ;       UCB$W_XA_CSRTMP(R5) = CSR image to start transfer with
                                0275
                                0275   ; Set DR11-W registers and start transfer
                                0275   ; Note that read-modify-write cycles are NOT performed to the DR11-W CSR.
                                0275   ; The CSR is always written directly into.  This prevents inadvertently setting
                                0275   ; the EIR select flag (writing bit 15) if error happens to become true.
                                0275
                                0275           DSBINT                           ; Disable interrupts (powerfail)
          64   00B4 C5    AE   027B           MNEGW   UCB$L_XA_DPR(R5),XA_WCR(R4)
```

```
                        0280   ; Load negative of transfer count
   02 A4   51   B0      0280          MOVW    R1,XA_BAR(R4)          ; Load low 16 bits of bus address
   04 A4   50   B0      0284          MOVW    R0,XA_CSR(R4)          ; Load CSR image less "GO" and "CYCLE"
06 44 A5   01   E1      0288          BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),26$ ; Link mode?
   04 A4   52   B0      028D          MOVW    R2,XA_CSR(R4)          ; Yes, load CSR image less "FNCT" bit 2
           06   11      0291          BRB     126$            ; Only if link mode in dev characteristics
                        0293  26$:
04 A4   00A4 C5   B0    0293          MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Move all bits to CSR
                        0299
                        0299  ; Wait for transfer complete interrupt, powerfail, or device time-out
                        0299
                        0299  126$:
                        0299          WFIKPCH XA_TIME_OUT,IRP$L_MEDIA(R3) ; Wait for interrupt
                        02A4
                        02A4  ; Device has interrupted, FORK
                        02A4
                        02A4          IOFORK                         ; FORK to lower IPL
                        02AA
                        02AA  ; Handle request completion, release UBA resources, check for errors
                        02AA
      7E   01   3C      02AA          MOVZWL  #SS$_NORMAL,-(SP)      ; Assume success, store code on stack
      00C0 C5   B4      02AD          CLRW    UCB$W_XA_DPRN(R5)      ; Clear DPR number and DPR error flag
                        02B1          PURDPR                         ; Purge UBA buffered data path
         09 50   E8     02B7          BLBS    R0,27$                 ; Branch if no datapath error
   6E    0154 8F   3C   02BA          MOVZWL  #SS$_PARITY,(SP)       ; Flag parity error on device
      00C1 C5   96      02BF          INCB    UCB$W_XA_DPRN+1(R5)    ; Flag PDR error for log
   00B4 C5   51   D0    02C3  27$:    MOVL    R1,UCB$L_XA_DPR(R5)    ; Save data path register in UCB
           00   EF      02C8          EXTZV   #VEC$V_DATAPATH,-      ; Get Datapath register no.
           05           02CA                  #VEC$S_DATAPATH,-      ; For Error Log
         50   37 A3     02CB                  CRB$L_INTD+VEC$B_DATAPATH(R3),R0
      00C0 C5   50   90 02CE          MOVB    R0,UCB$W_XA_DPRN(R5)   ; Save for later in UCB
50    00AE C5   07   09 EF  02D3     EXTZV   #9,#7,UCB$W_XA_BAR(R5),R0 ; Low bits, final map register no.
51    00A8 C5   02   04 EF  02DA     EXTZV   #4,#2,UCB$W_XA_CSR(R5),R1 ; Hi bits of map register no.
      50   02   07   51 F0  02E1     INSV    R1,#7,#2,R0            ; Entire map register number
      01F0 8F   50   B1 02E6         CMPW    R0,#496                ; Is map register number in range?
           1A   14   02EB          BGTR    28$                    ; No, forget it - compound error
   00B8 C5   6240   D0 02ED         MOVL    (R2)[R0],UCB$L_XA_FMPR(R5) ; Save map register contents
      00BC C5   D4   02F3          CLRL    UCB$L_XA_PMPR(R5)      ; Assume no previous map register
           50   D7   02F7          DECL    R0                     ; Was there a previous map register?
      0F   00   EC   02F9          CMPV    #VEC$V_MAPREG,#VEC$S_MAPREG,-
         50   34 A3     02FC                  CRB$L_INTD+VEC$W_MAPREG(R3),R0
           06   14   02FF          BGTR    28$                    ; No if gtr
   00B8 C5   6240   D0 0301         MOVL    (R2)[R0],UCB$L_XA_FMPR(R5) ; Save previous map register contents
                        0307  28$:    RELMPR                         ; Release UBA resources
                        030D          RELDPR
                        0313
                        0313  ; Check for errors and return status
                        0313
      00B0 C5   B5      0313          TSTW    UCB$W_XA_WCR(R5)       ; All words transferred?
           05   13      0317          BEQL    30$                    ; Yes
   6E    02D4 8F   3C   0319          MOVZWL  #SS$_OPINCOMPL,(SP)    ; No, flag operation not complete
08 00A8 C5   0F   E1    031E  30$:    BBC     #XA_CSR$V_ERROR,UCB$W_XA_CSR(R5),35$ ; Branch on CSR error bit
   6E    00B2 C5   3C   0324          MOVZWL  UCB$W_XA_ERROR(R5),(SP) ; Flag for controller/drive error status
         02D2   30      0329          BSBW    XA_DEV_RESET           ; Reset DR11-W
      06 6E     E8      032C  35$:    BLBS    (SP),40$               ; Any errors after all this?
   00000000'GF   16     032F          JSB     G^ERL$DEVICERR         ; Yes, log them
         0248   30      0335  40$:    BSBW    DEL_ATTNAST            ; Deliver outstanding ATTN AST's
```

```
        00000000'GF   16  0338   729        JSB     G^IOC$DIAGBUFILL          ; Fill diagnostic buffer
             50   8E  D0  033E   730        MOVL    (SP)+,R0                  ; Get final device status
51   00B0 C5   02   A5  0341   731        MULW3   #2,UCB$W_XA_WCR(R5),R1    ; Calculate final transfer count
        51   7E A5   A0  0347   732        ADDW    UCB$W_BCNT(R5),R1
50   10   10   51   F0  034B   733        INSV    R1,#16,#16,R0             ; Insert into high byte of IOSB
     51   00A8 C5   D0  0350   734        MOVL    UCB$W_XA_CSR(R5),R1       ; Return CSR and EIR in IOSB
     04 A4   40 8F   88  0355   735        BISB    #XA_CSR$M_IE,XA_CSR(R4)   ; Enable interrupts
                     035A   736        REQCOM                                ; Finish request in exec
```

```
                        0360
                        0360            .DSABL  LSB
                        0360    ;++
                        0360    ; WORD MODE -- Process word mode (interrupt per word) transfer
                        0360    ;
                        0360    ; FUNCTIONAL DESCRIPTION:
                        0360    ;
                        0360    ;       Data is transferred one word at a time with an interrupt for each word.
                        0360    ;       The request is handled separately for a write (from memory to DR11-W
                        0360    ;       and a read (from DR11-W to memory).
                        0360    ;       For a write, data is fetched from memory, loaded into the ODR of the
                        0360    ;       DR11-W and the system waits for an interrupt.  For a read, the system
                        0360    ;       waits for a DR11-W interrupt and the IDR is transferred into memory.
                        0360    ;       If the unsolicited interrupt flag is set, the first word is transferred
                        0360    ;       directly into memory withou waiting for an interrupt.
                        0360    ;--
                        0360
                        0360            .ENABL  LSB
                        0360    WORD_MODE:
                        0360
                        0360
                        0360    ; Dispatch to separate loops on READ or WRITE
                        0360
52   0C   91            0360            CMPB    #IO$_READPBLK,R2        ; Check for read function
     6E   13            0363            BEQL    30$
                        0365
                        0365    ;++
                        0365    ; WORD MODE WRITE -- Write (output) in word mode
                        0365    ;
                        0365    ; FUNCTIONAL DESCRIPTION:
                        0365    ;
                        0365    ;       Transfer the requested number of words from user memory to
                        0365    ;       the DR11-W ODR one word at a time, wait for interrupt for each
                        0365    ;       word.
                        0365    ;--
                        0365
                        0365    10$:
          00DA      30  0365            BSBW    MOVFRUSER               ; Get two bytes from user buffer
                        0368            DSBINT                          ; Lock out interrupts
                        036E                                            ; Flag interrupt expected
        06 A4   51  B0  036E            MOVW    R1,XA_ODR(R4)           ; Move data to DR11-W
     04 A4   00A4 C5  B0  0372          MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Set DR11-W CSR
        07 44 A5     01  E1  0378       BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),15$ ; Link mode?
04 A4   00A4 C5      04  AB  037D       BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Clear interrupt FNCT bi
                        0384                                            ; Only if link mode specified
                        0384    15$:
                        0384
                        0384    ; Wait for interrupt, powerfail, or device time-out
                        0384
                        0384            WFIKPCH XA_TIME_OUTW,IRP$L_MEDIA(R3)
                        038F
                        038F    ; Check for errors, decrement transfer count, and loop til complete
                        038F
                        038F            IOFORK                          ; Fork to lower IPL
 00AA C5   5E00 8F  B3  0395            BITW    #XA_EIR$M_NEX!-
                        039C                    XA_EIR$M_MULTI!-
                        039C                    XA_EIR$M_ACLO!-
```

```
                        039C                    XA_EIR$M_PAR!-
                        039C                    XA_EIR$M_DLT,UCB$W_XA_EIR(R5) ; Any errors?
                 03   13 039C            BEQL    20$                           ; No, continue
               0087   31 039E            BRW     40$                           ; Yes, abort transfer.
           00B4 C5    B7 03A1     20$:   DECW    UCB$L_XA_DPR(R5)                    ; All words trnasferred?
                BE    12 03A5            BNEQ    10$                           ; No, loop until finished.
                        03A7
                        03A7 ; Transfer is done, clear iterrupt expected flag and FORK
                        03A7 ; All words read or written in WORD MODE.  Finish I/O.
                        03A7
                        03A7 RETURN_STATUS:
                        03A7
       00000000'GF   16 03A7            JSB     G^IOC$DIAGBUFILL              ; Fill diagnostic buffer if present
             01DC    30 03AD            BSBW    DEL_ATTNAST                   ; Deliver outstanding ATTN AST's
            50   01    3C 03B0            MOVZWL  #SS$_NORMAL,R0                ; Complete success status
51   00B4 C5   02    A5 03B3     22$:   MULW3   #2,UCB$L_XA_DPR(R5),R1        ; Calculate actual bytes xfered
  51    7E A5   51    A3 03B9            SUBW3   R1,UCB$W_BCNT(R5),R1          ; From requested number of bytes
50   10   10   51    F0 03BE            INSV    R1,#16,#16,R0                 ; And place in high word of R0
     51   00A8 C5    D0 03C3            MOVL    UCB$W_XA_CSR(R5),R1           ; Return CSR and EIR status
     04 A4   40 8F    88 03C8            BISB    #XA_CSR$M_IE,XA_CSR(R4)       ; Enable device interrupts
                        03CD            REQCOM                                ; Finish request in exec
                        03D3
                        03D3 ;++
                        03D3 ; WORD MODE READ -- Read (input) in word mode
                        03D3 ;
                        03D3 ; FUNCTIONAL DESCRIPTION:
                        03D3 ;
                        03D3 ;          Transfer the requested number of wrods from the DR11-W IDR into
                        03D3 ;          user memory one word at a time, wait for interrupt for each word.
                        03D3 ;          If the unexpected (unsolicited) interrupt bit is set, transfer the
                        03D3 ;          first (last received) word to memory without waiting for an
                        03D3 ;          interrupt.
                        03D3 ;--
                        03D3
                        03D3 30$:
                        03D3            DSBINT  UCB$B_DIPL(R5)                ; Lock out interrupts
                        03DA
                        03DA ; If an unexpected (unsolicited) interrupt has occured, assume it
                        03DA ; is for this READ request and return value to user buffer without
                        03DA ; waiting for an interrupt.
                        03DA
                 01   E5 03DA            BBCC    #UCB$V_UNEXPT,-
           05 68 A5     03DC                    UCB$W_DEVSTS(R5),32$          ; Branch if no unexpected interrupt
                        03DF            ENBINT                                ; Enable interrupts
                 14   11 03E2            BRB     37$                           ; continue
                        03E4
                        03E4 32$:
                        03E4            SETIPL  #IPL$_POWER
                        03E7 35$:
                        03E7
                        03E7 ; Wait for interrupt, powerfail, or device time-out
                        03E7
                        03E7            WFIKPCH XA_TIME_OUTW,IRP$L_MEDIA(R3)
                        03F2
                        03F2 ; Check for errors, decrement transfer count and loop until done
                        03F2
                        03F2            IOFORK                                ; Fork to lower IPL
```

```
                                03F8   852 37$:
00AA C5   5E00 8F   B3          03F8   853          BITW    #XA_EIR$M_NEX!-
                                03FF   854                  XA_EIR$M_MULTI!-
                                03FF   855                  XA_EIR$M_ACLO!-
                                03FF   856                  XA_EIR$M_PAR!-
                                03FF   857                  XA_EIR$M_DLT,UCB$W_XA_EIR(R5) ; Any errors?
                     27   12    03FF   858          BNEQ    40$                           ; Yes, abort transfer.
                   004F   30    0401   859          BSBW    MOVTOUSER                     ; Store two bytes into user buffer
                                0404   860
                                0404   861 ; Send interrupt back to sender.  Acknowledge we got last word.
                                0404   862
                                0404   863          DSBINT
      04 A4    00A4 C5   B0     040A   864          MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4)
         07 44 A5      01   E1  0410   865          BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),38$ ; Link mode?
04 A4    00A4 C5      04   AB  0415   866          BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Yes, clear FNCT 2
                                041C   867 38$:
             00B4 C5   B7       041C   868          DECW    UCB$L_XA_DPR(R5)                ; Decrement transfer count
                  C5   12       0420   869          BNEQ    35$                           ; Loop until all words transferred
                                0422   870          ENBINT
                   FF7F   31    0425   871          BRW     RETURN_STATUS                 ; Finish request in common code
                                0428   872
                                0428   873 ; Error detected in word mode transfer
                                0428   874
                                0428   875 40$:
                   0155   30    0428   876          BSBW    DEL_ATTNAST                   ; Deliver ATTN AST's
                   01D0   30    042B   877          BSBW    XA_DEV_RESET                  ; Error, reset DR11-W
            00000000'GF   16    042E   878          JSB     G^IOC$DIAGBUFILL              ; Fill diagnostic buffer if presetn
            00000000'GF   16    0434   879          JSB     G^ERL$DEVICERR                ; Log device error
         50    00B2 C5   3C    043A   880          MOVZWL  UCB$W_XA_ERROR(R5),R0         ; Set controller/drive status in R0
                   FF71   31    043F   881          BRW     22$
                                0442   882
                                0442   883          .DSABL  LSB
                                0442   884 ;
                                0442   885 ; MOVFRUSER - Routine to fetch two bytes from user buffer.
                                0442   886 ;
                                0442   887 ; INPUTS:
                                0442   888 ;
                                0442   889 ;        R5 = UCB address
                                0442   890 ;
                                0442   891 ; OUTPUTS:
                                0442   892 ;
                                0442   893 ;        R1 = Two bytes of data from users buffer
                                0442   894 ;        Buffer descriptor in UCB is updated.
                                0442   895 ;
                                0442   896          .ENABL  LSB
                                0442   897 MOVFRUSER:
         51   7E   DE            0442   898          MOVAL   -(SP),R1                      ; Address of temporary stack loc
         52   02   9A            0445   899          MOVZBL  #2,R2                         ; Fetch two bytes
   00000000'GF   16              0448   900          JSB     G^IOC$MOVFRUSER               ; Call exec routine to do the deed
         51   8E   D0            044E   901          MOVL    (SP)+,R1                      ; Retreive the bytes
              0E   11            0451   902          BRB     20$                           ; Update UCB buffer pointers
                                0453   903 ;
                                0453   904 ; MOVTOUSER - Routine to store two bytes into users buffer.
                                0453   905 ;
                                0453   906 ; INPUTS:
                                0453   907 ;
                                0453   908 ;        R5 = UCB address
```

```
                        0453   ;       UCB$W_XA_IDR(R5) = Location where two bytes are saved
                        0453   ;
                        0453   ; OUTPUTS:
                        0453   ;
                        0453   ;       Two bytes are stored in user buffer and buffer descriptor in
                        0453   ;       UCB is updated.
                        0453   ;
                        0453   MOVTOUSER:
   51   00AC C5   9E    0453           MOVAB   UCB$W_XA_IDR(R5),R1     ; Address of internal buffer
        52   02   9A    0458           MOVZBL  #2,R2
 00000000'GF    16      045B           JSB     G^IOC$MOVTOUSER         ; Call exec
                        0461   20$:                                    ; Update buffer pointers in UCB
     7C A5   02   A0    0461           ADDW    #2,UCB$W_BOFF(R5)       ; Add two to buffer descriptor
7C A5   FE00 8F   AA    0465           BICW    #^C<^X01FF>,UCB$W_BOFF(R5) ; Modulo the page size
             04   12    046B           BNEQ    30$                     ; If NEQ, no page boundary crossed
     78 A5   04   C0    046D           ADDL    #4,UCB$L_SVAPTE(R5)     ; Point to next page
                        0471   30$:
             05         0471           RSB
                        0472   ;
                        0472           .DSABL  LSB
```

```
                        0472          .SBTTL  DR11-W DEVICE TIME-OUT
                        0472  ;++
                        0472  ; DR11-W device TIME-OUT
                        0472  ; If a DMA transfer was in progress, release UBA resources.
                        0472  ; For DMA or WORD mode, deliver ATTN AST's, log a device timeout error,
                        0472  ; and do a hard reset on the controller.
                        0472  ;
                        0472  ; Clear DR11-W CSR
                        0472  ; Return error status
                        0472  ;
                        0472  ; Power failure will appear as a device time-out
                        0472  ;--
                        0472          .ENABL  LSB
                        0472  XA_TIME_OUT:                            ; Time-out for DMA transfer
                        0472
                        0472          SETIPL  UCB$B_FIPL(R5)          ; Lower to FORK IPL
                        0476          PURDPR                          ; Purge buffered data path in UBA
                        047C          RELMPR                          ; Release UBA map registers
                        0482          RELDPR                          ; Release UBA data path
                 04 11  0488          BRB     10$                     ; continue
                        048A
                        048A  XA_TIME_OUTW:                           ; Time-out for WORD mode transfer
                        048A
                        048A          SETIPL  UCB$B_FIPL(R5)          ; Lower to FORK IPL
              54 24 A5  D0  048E  10$:  MOVL    UCB$L_CRB(R5),R4        ; Fetch address of CSR
              54 2C B4  D0  0492          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                   005C 30  0496          BSBW    XA_REGISTER             ; Read DR11-W registers
           00000000'GF 16  0499          JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer
           00000000'GF 16  049F          JSB     G^ERL$DEVICTMO          ; Log device time out
                   00D8 30  04A5          BSBW    DEL_ATTNAST             ; And deliver the AST's
                   0153 30  04A8          BSBW    XA_DEV_RESET            ; Reset controller
           50 022C 8F   3C  04AB          MOVZWL  #SS$_TIMEOUT,R0         ; Assume error status
                     03 E1  04B0          BBC     #UCB$V_CANCEL,-
              05 64 A5      04B2                  UCB$W_STS(R5),20$       ; Branch if not cancel
           50 0830 8F   3C  04B5          MOVZWL  #SS$_CANCEL,R0          ; Set status
                     51 D4  04BA  20$:    CLRL    R1
                  68 A5 B4  04BC          CLRW    UCB$W_DEVSTS(R5)        ; Clear ATTN AST flags
                006B 8F AA  04BF          BICW    #<UCB$M_TIM!UCB$M_INT!UCB$M_TIMOUT!UCB$M_CANCEL!UCB$M_POWER>,-
                  64 A5     04C3                  UCB$W_STS(R5)           ; Clear unit status flags
                            04C5          REQCOM                          ; Complete I/O in exec
                            04CB          .DSABL  LSB
```

```
                    04CB        .SBTTL  XA_INTERRUPT, Interrupt service routine for DR11-W
                    04CB  ;++
                    04CB  ; XA_INTERRUPT, Handles interrupts generated by DR11-W
                    04CB  ;
                    04CB  ; Functional description:
                    04CB  ;
                    04CB  ;       This routine is entered whenever an interrupt is generated
                    04CB  ;       by the DR11-W.  It checks that an interrupt was expected.
                    04CB  ;       If not, it sets the unexpected (unsolicited) interrupt flag.
                    04CB  ;       All device registers are read and stored into the UCB.
                    04CB  ;       If an interrupt was expected, it calls the driver back at its Wait
                    04CB  ;       For Interrupt point.
                    04CB  ;       Deliver ATTN AST's if unexpected interrupt.
                    04CB  ;
                    04CB  ; Inputs:
                    04CB  ;
                    04CB  ;       00(SP) = Pointer to address of the device IDB
                    04CB  ;       04(SP) = saved R0
                    04CB  ;       08(SP) = saved R1
                    04CB  ;       12(SP) = saved R2
                    04CB  ;       16(SP) = saved R3
                    04CB  ;       20(SP) = saved R4
                    04CB  ;       24(SP) = saved R5
                    04CB  ;       28(SP) = saved PSL
                    04CB  ;       32(SP) = saved PC
                    04CB  ;
                    04CB  ; Outputs:
                    04CB  ;
                    04CB  ;       The driver is called at its Wait For Interrupt point if an
                    04CB  ;       interrupt was expected.
                    04CB  ;       The current value of the DR11-W CSR's are stored in the UCB.
                    04CB  ;
                    04CB  ;--
                    04CB  XA_INTERRUPT:                                   ; Interrupt service for DR11-W
        54    9E    D0  04CB          MOVL    @(SP)+,R4                   ; Address of IDB and pop SP
        54    64    7D  04CE          MOVQ    (R4),R4                     ; CSR and UCB address from IDB
                        04D1
                        04D1  ; Read the DR11-W device registers (WCR, BAR, CSR, EIR, IDR) and store
                        04D1  ; into UCB.
                        04D1
              0021    30  04D1          BSBW    XA_REGISTER                 ; Read device registers
                        04D4
                        04D4  ; Check to see if device transfer request active or not
                        04D4  ; If so, call driver back at Wait for Interrupt point and
                        04D4  ; Clear unexpected interrupt flag.
                        04D4
  0D 64 A5    01    E5  04D4  20$:    BBCC    #UCB$V_INT,UCB$W_STS(R5),25$
                        04D9                                              ; If clear, no interrupt expected
                        04D9
                        04D9  ; Interrupt expected, clear unexpected interrupt flag and call driver
                        04D9  ; back.
                        04D9
     68 A5    02    AA  04D9          BICW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R5)
                        04DD                                              ; Clear unexpected interrupt flag
     53    10 A5    D0  04DD          MOVL    UCB$L_FR3(R5),R3            ; Restore drivers R3
           0C B5    16  04E1          JSB     @UCB$L_FPC(R5)              ; Call driver back
              0C    11  04E4          BRB     30$
```

```
                    04E6  1031
                    04E6  1032 ; Deliver ATTN AST's if no interrupt expected and set unexpected
                    04E6  1033 ; interrupt flag.
                    04E6  1034
                    04E6  1035 25$:
   68 A5    02  A8  04E6  1036          BISW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R5) ; Set unexpected interrupt flag
         0093  30  04EA  1037          BSBW    DEL_ATTNAST                    ; Deliver ATTN AST's
04 A4    40 8F  88  04ED  1038          BISB    #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts
                    04F2  1039
                    04F2  1040 ; Restore registers and return from interrupt
                    04F2  1041
                    04F2  1042 30$:
            3F  BA  04F2  1043          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
            02      04F4  1044          REI                             ; Return from interrupt
```

```
                        04F5  1047          .SBTTL  XA_REGISTER - Handle DR11-W CSR transfers
                        04F5  1048 ;++
                        04F5  1049 ; XA_REGISTER - Routine to handle DR11-W register transfers
                        04F5  1050 ;
                        04F5  1051 ; INPUTS:
                        04F5  1052 ;
                        04F5  1053 ;        R4 - DR11-W CSR address
                        04F5  1054 ;        R5 - UCB address of unit
                        04F5  1055 ;
                        04F5  1056 ; OUTPUTS:
                        04F5  1057 ;
                        04F5  1058 ;        CSR, EIR, WCR, BAR, IDR, and status are read and stored into UCB.
                        04F5  1059 ;        The DR11-W is placed in its initial state with interrupts enabled.
                        04F5  1060 ;        R0 - .true. if no hard error
                        04F5  1061 ;             .false. if hard error (cannot clear ATTN)
                        04F5  1062 ;
                        04F5  1063 ; If the CSR ERROR bit is set and the associated condition can be cleared, then
                        04F5  1064 ; the error is transient and recoverable.  The status returned is SS$_DRVERR.
                        04F5  1065 ; If the CSR ERROR bit is set and cannot be cleared by clearing the CSR, then
                        04F5  1066 ; this is a hard error and cannot be recovered.  The returned status is
                        04F5  1067 ; SS$_CTRLERR.
                        04F5  1068 ;
                        04F5  1069 ;        R0,R1 - destroyed, all other registers preserved.
                        04F5  1070 ;--
                        04F5  1071
                        04F5  1072 XA_REGISTER:
                        04F5  1073
          50    01   3C 04F5  1074          MOVZWL  #SS$_NORMAL,R0          ; Assume success
       51    04 A4   3C 04F8  1075          MOVZWL  XA_CSR(R4),R1           ; Read CSR
    00A8 C5    51   B0 04FC  1076          MOVW    R1,UCB$W_XA_CSR(R5)     ; Save CSR in UCB
       05 51    0F   E1 0501  1077          BBC     #XA_CSR$V_ERROR,R1,55$  ; Branch if no error
    50    008C 8F   3C 0505  1078          MOVZWL  #SS$_DRVERR,R0          ; Assume "drive" error
    51    FFF1 8F   AA 050A  1079 55$:      BICW    #^C<XA_CSR$M_FNCT>,R1   ; Clear all uninteresting bits for later
  05 A4    80 8F   88 050F  1080          BISB    #<XA_CSR$M_ERROR/256>,XA_CSR+1(R4) ; Set EIR flag
00AA C5    04 A4   B0 0514  1081          MOVW    XA_EIR(R4),UCB$W_XA_EIR(R5) ; Save EIR in UCB
     04 A4    51   B0 051A  1082          MOVW    R1,XA_CSR(R4)           ; Clear EIR flag and errors
     51    04 A4   B0 051E  1083          MOVW    XA_CSR(R4),R1           ; Read CSR back
     05 51    0D   E1 0522  1084          BBC     #XA_CSR$V_ATTN,R1,60$   ; If attention still set, hard error
    50    0054 8F   3C 0526  1085          MOVZWL  #SS$_CTRLERR,R0         ; Flag hard controller error
00AC C5    06 A4   B0 052B  1086 60$:      MOVW    XA_IDR(R4),UCB$W_XA_IDR(R5) ; Save IDR in UCB
00AE C5    02 A4   B0 0531  1087          MOVW    XA_BAR(R4),UCB$W_XA_BAR(R5)
    00B0 C5    64   B0 0537  1088          MOVW    XA_WCR(R4),UCB$W_XA_WCR(R5)
    00B2 C5    50   B0 053C  1089          MOVW    R0,UCB$W_XA_ERROR(R5)   ; Save status in UCB
                    05 0541  1090          RSB
```

```
                        0542  1092          .SBTTL  XA_CANCEL, Cancel I/O routine
                        0542  1093 ;++
                        0542  1094 ; XA_CANCEL, Cancels an I/O operation in progress
                        0542  1095 ;
                        0542  1096 ; Functional description:
                        0542  1097 ;
                        0542  1098 ;       Flushes Attention AST queue for the user.
                        0542  1099 ;       If transfer in progress, do a device reset to DR11-W and finish the
                        0542  1100 ;       request.
                        0542  1101 ;       Clear interrupt expected flag.
                        0542  1102 ;
                        0542  1103 ; Inputs:
                        0542  1104 ;
                        0542  1105 ;       R2 = negated value of channel index
                        0542  1106 ;       R3 = address of current IRP
                        0542  1107 ;       R4 = address of the PCB requesting the cancel
                        0542  1108 ;       R5 = address of the device's UCB
                        0542  1109 ;
                        0542  1110 ; Outputs:
                        0542  1111 ;
                        0542  1112 ;--
                        0542  1113
                        0542  1114 XA_CANCEL:                                      ; Cancel I/O
                        0542  1115
          00    E5      0542  1116          BBCC    #UCB$V_ATTNAST,-
       16 68 A5         0544  1117                  UCB$W_DEVSTS(R5),20$     ; ATTN AST enabled?
                        0547  1118
                        0547  1119 ; Finish all ATTN AST's for this process.
                        0547  1120
       00C4 8F    BB    0547  1121          PUSHR   #^M<R2,R6,R7>
          56      52    D0    054B  1122          MOVL    R2,R6                    ; Set up channel number
   57  00A0 C5          9E    054E  1123          MOVAB   UCB$L_XA_ATTN(R5),R7     ; Address of listhead
 00000000'GF            16    0553  1124          JSB     G^COM$FLUSHATTNS         ; Flush ATTN AST's for process
       00C4 8F    BA    0559  1125          POPR    #^M<R2,R6,R7>
                        055D  1126
                        055D  1127 ; Check to see if a data transfer request is in progress
                        055D  1128 ; for this process on this channel
                        055D  1129
                        055D  1130 20$:
                        055D  1131          DSBINT  UCB$B_DIPL(R5)           ; Lock out device interrupts
 00000000'GF            16    0564  1132          JSB     G^IOC$CANCELIO           ; Check if transfer going
          03    E1      056A  1133          BBC     #UCB$V_CANCEL,-
       0D 64 A5         056C  1134                  UCB$W_STS(R5),30$        ; Branch if not for this guy
                        056F  1135 ;
                        056F  1136 ; Force timeout
                        056F  1137 ;
          6C A5    D4    056F  1138          CLRL    UCB$L_DUETIM(R5)         ; clear timer
       64 A5    01    A8    0572  1139          BISW    #UCB$M_TIM,UCB$W_STS(R5) ; set timed
       0040 8F    AA    0576  1140          BICW    #UCB$M_TIMOUT,-
          64 A5         057A  1141                  UCB$W_STS(R5)            ; Clear timed out
                        057C  1142 30$:
                        057C  1143          ENBINT                           ; Lower to FORK IPL
                 05     057F  1144          RSB                              ; Return
```

```
                                0580          .SBTTL  DEL_ATTNAST, Deliver ATTN AST's
                                0580  ;++
                                0580  ; DEL_ATTNAST, Deliver all outstanding ATTN AST's
                                0580  ;
                                0580  ; Functional description:
                                0580  ;
                                0580  ;       This routine is used by the DR11-W driver to deliver all of the
                                0580  ;       outstanding attention AST's.  It is copied from COM$DELATTNAST in
                                0580  ;       the exec.  In addition, it places the saved value of the DR11-W CSR
                                0580  ;       and Input Data Buffer Register in the AST paramater.
                                0580  ;
                                0580  ; Inputs:
                                0580  ;
                                0580  ;       R5 = UCB of DR11-W unit
                                0580  ;
                                0580  ; Outputs:
                                0580  ;
                                0580  ;       R0,R1,R2 Destroyed
                                0580  ;       R3,R4,R5 Preserved
                                0580  ;--
                                0580  DEL_ATTNAST:
                                0580          DSBINT  UCB$B_DIPL(R5)          ; Device IPL
   49 68 A5    00    E5         0587          BBCC    #UCB$V_ATTNAST,UCB$W_DEVSTS(R5),30$
                                058C                                          ; Any ATTN AST's expected?
               38    BB         058C          PUSHR   #^M<R3,R4,R5>           ; Save R3,R4,R5
      51       08 AE  D0         058E  10$:    MOVL    8(SP),R1                ; Get address of UCB
   52       00A0 C1  9E         0592          MOVAB   UCB$L_XA_ATTN(R1),R2    ; Address of ATTN AST listhead
            55    62  D0         0597          MOVL    (R2),R5                 ; Address of next entry on list
               37    13         059A          BEQL    20$                     ; No next entry, end of loop
      68 A1    02    AA         059C          BICW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R1) ; Clear unexpected interrupt flag
            62    65  D0         05A0          MOVL    (R5),(R2)               ; Close list
1E A5       00AC C1  B0         05A3          MOVW    UCB$W_XA_IDR(R1),ACB$L_KAST+6(R5)
                                05A9                                          ; Store IDR in AST paramater
1C A5       00A8 C1  B0         05A9          MOVW    UCB$W_XA_CSR(R1),ACB$L_KAST+4(R5)
                                05AF                                          ; Store CSR in AST paramater
            DC AF    9F         05AF          PUSHAB  B^10$                   ; Set return address for FORK
                                05B2          FORK                            ; FORK for this AST
                                05B8
                                05B8  ; AST fork procedure
                                05B8
10 A5       18 A5    7D         05B8          MOVQ    ACB$L_KAST(R5),ACB$L_AST(R5)
                                05BD                                          ; Re-arrange entries
0B A5       20 A5    90         05BD          MOVB    ACB$L_KAST+8(R5),ACB$B_RMOD(R5)
0C A5       24 A5    D0         05C2          MOVL    ACB$L_KAST+12(R5),ACB$L_PID(R5)
            18 A5    D4         05C7          CLRL    ACB$L_KAST(R5)
         52    01    9A         05CA          MOVZBL  #PRI$_IOCOM,R2          ; Set up priority increment
   00000000'GF       17         05CD          JMP     G^SCH$QAST              ; Queue the AST
                                05D3
               38    BA         05D3  20$:    POPR    #^M<R3,R4,R5>           ; Restore registers
                                05D5  30$:    ENBINT                          ; Enable interrupts
                  05            05D8          RSB                             ; Return
```

```
                05D9  .SBTTL  XA_REGDUMP - DR11-W register dump routine
                05D9 ;++
                05D9 ; XA_REGDUMP - DR11-W Register dump routine.
                05D9 ;
                05D9 ; This routine is called to save the controller registers in a specified
                05D9 ; buffer.  It is called from the device error logging routine and from the
                05D9 ; diagnostic buffer fill routine.
                05D9 ;
                05D9 ; Inputs:
                05D9 ;
                05D9 ;       R0 - Address of register save buffer
                05D9 ;       R4 - Address of Control and Status Register
                05D9 ;       R5 - Address of UCB
                05D9 ;
                05D9 ; Outputs:
                05D9 ;
                05D9 ;       The controller registers are saved in the specified buffer.
                05D9 ;
                05D9 ;               CSRTMP - The last command written to the DR11-W CSR by
                05D9 ;                        by the driver.
                05D9 ;               BARTMP - The last value written into the DR11-W BAR by
                05D9 ;                        the driver during a block mode transfer.
                05D9 ;               CSR - The CSR image at the last interrupt
                05D9 ;               EIR - The EIR image at the last interrupt
                05D9 ;               IDR - The IDR image at the last interrupt
                05D9 ;               BAR - The BAR image at the last interrupt
                05D9 ;               WCR - Word count register
                05D9 ;               ERROR - The system status at request completion
                05D9 ;               PDRN - UBA Datapath Register number
                05D9 ;               DPR - The contents of the UBA Data Path register
                05D9 ;               FMPR - The contents of the last UBA Map register
                05D9 ;               PMRP - The contents of the previous UBA Map register
                05D9 ;               DPRF - Flag for purge datapath error
                05D9 ;                      0 = no purger datapath error
                05D9 ;                      1 = parity error when datapath was purged
                05D9 ;
                05D9 ;       Note that the values stored are from the last completed transfer
                05D9 ;       operation. If a zero transfer count is specified, then the
                05D9 ;       values are from the last operation with a non-zero transfer count.
                05D9 ;--
                05D9
                05D9 XA_REGDUMP:
                05D9
        80   0B  9A  05D9          MOVZBL  #11,(R0)+                  ; Eleven registers are stored.
51      00A4 C5  9E  05DC          MOVAB   UCB$W_XA_CSRTMP(R5),R1     ; Get address of saved register images
        52   08  9A  05E1          MOVZBL  #8,R2                      ; Return 8 registers here
        80   81  3C  05E4  10$:    MOVZWL  (R1)+,(R0)+
           FA 52  F5  05E7          SOBGTR  R2,10$                     ; Move them all
80      00C0 C5  9A  05EA          MOVZBL  UCB$W_XA_DPRN(R5),(R0)+    ; Save Datapath Register number
        52   03  9A  05EF          MOVZBL  #3,R2                      ; And 3 more here
        80   81  D0  05F2  20$:    MOVL    (R1)+,(R0)+                ; Move UBA register contents
           FA 52  F5  05F5          SOBGTR  R2,20$
80      00C1 C5  9A  05F8          MOVZBL  UCB$W_XA_DPRN+1(R5),(R0)+  ; Save Datapath Parity Error Flag
                 05  05FD          RSB
```

```
                        05FE          .SBTTL  XA_DEV_RESET - Device reset DR11-W
                        05FE  ;++
                        05FE  ; XA_DEV_RESET - DR11-W Device reset routine
                        05FE  ;
                        05FE  ; This routine raises IPL to device IPL, performs a device reset to
                        05FE  ; the required controler, and re-enables device interrupts.
                        05FE  ;
                        05FE  ; Inputs:
                        05FE  ;
                        05FE  ;       R4 - Address of Control and Status Register
                        05FE  ;       R5 - Address of UCB
                        05FE  ;
                        05FE  ; Outputs:
                        05FE  ;
                        05FE  ;       Controller is reset, controller interrupts are enabled
                        05FE  ;
                        05FE  ;--
                        05FE
                        05FE  XA_DEV_RESET:
                        05FE
                 07  BB 05FE          PUSHR   #^M<R0,R1,R2>           ; Save some registers
                        0600          DSBINT                          ; Raise IPL to lock all interrupts
           05 A4    10  90 0606          MOVB    #<XA_CSR$M_MAINT/256>,XA_CSR+1(R4)
              05 A4    94 060A          CLRB    XA_CSR+1(R4)
                        060D
                        060D  ; ***  Must delay here depending on reset interval
                        060D
                        060D          TIMEDWAIT TIME=#XA_RESET_DELAY  ; No. of 10 micro-sec intervals to wait
                        062B
     04 A4    40 8F    90 062B          MOVB    #XA_CSR$M_IE,XA_CSR(R4) ; Re-enable device interrupts
                        0630          ENBINT                          ; Restore IPL
                 07  BA 0633          POPR    #^M<R0,R1,R2>           ; Restore registers
                        0635
                    05  0635          RSB
                        0636
                        0636  XA_END:                                         ; End of driver label
                        0636          .END
```

```
$$$                     = 00000020 R     02
$$OP                    = 00000002
ACB$B_RMOD              = 0000000B
ACB$L_AST               = 00000010
ACB$L_KAST              = 00000018
ACB$L_PID               = 0000000C
AT$_UBA                 = 00000001
BLOCK_MODE                00000226 R     03
COM$F[USHATTNS            ********   X   03
COM$SETATTNAST            ********   X   03
CRB$L_INTD              = 00000024
DC$_REALTIME            = 00000060
DDB$L_DDT               = 0000000C
DEL_ATTNAST               00000580 R     03
DEV$M_AVL               = 00040000
DEV$M_ELG               = 00400000
DEV$M_IDV               = 04000000
DEV$M_ODV               = 08000000
DEV$M_RTM               = 20000000
DPT$C_LENGTH            = 00000038
DPT$C_VERSION           = 00000004
DPT$INITAB                00000038 R     02
DPT$M_SVP               = 00000002
DPT$REINITAB              00000054 R     02
DPT$TAB                   00000000 R     02
DT$_DR11W               = 00000004
DYN$C_CRB               = 00000005
DYN$C_DDB               = 00000006
DYN$C_DPT               = 0000001E
DYN$C_UCB               = 00000010
EMB$L_DV_REGSAV         = 0000004E
ERL$DEVICERR              ********   X   03
ERL$DEVICTMO              ********   X   03
EXE$ABORTIO               ********   X   03
EXE$FINISHIOC             ********   X   03
EXE$FORK                  ********   X   03
EXE$GL_TENUSEC            ********   X   03
EXE$GL_UBDELAY            ********   X   03
EXE$IOFORK                ********   X   03
EXE$MODIFY                ********   X   03
EXE$READ                  ********   X   03
EXE$SENSEMODE             ********   X   03
EXE$SETCHAR               ********   X   03
EXE$SETMODE               ********   X   03
EXE$WRITE                 ********   X   03
FUNCTAB_LEN             = 0000004C
IDB$L_CSR               = 00000000
IDB$L_OWNER             = 00000004
IDB$L_UCBLST            = 00000018
IO$S_FCODE              = 00000006
IO$V_ATTNAST            = 00000008
IO$V_CYCLE              = 0000000C
IO$V_DATAPATH           = 0000000A
IO$V_DIAGNOSTIC         = 00000008
IO$V_FCODE              = 00000000
IO$V_RESET              = 0000000B
IO$V_SETFNCT            = 00000009
IO$V_TIMED              = 00000007
IO$V_WORD               = 00000006
IO$_READLBLK            = 00000021
IO$_READPBLK            = 0000000C
IO$_READVBLK            = 00000031
IO$_SENSECHAR           = 0000001B
IO$_SENSEMODE           = 00000027
IO$_SETCHAR             = 0000001A
IO$_SETMODE             = 00000023
IO$_VIRTUAL             = 0000003F
IO$_WRITELBLK           = 00000020
IO$_WRITEPBLK           = 0000000B
IO$_WRITEVBLK           = 00000030
IOC$CANCELIO              ********   X   03
IOC$DIAGBUFILL            ********   X   03
IOC$LOADUBAMAP            ********   X   03
IOC$MNTVER                ********   X   03
IOC$MOVFRUSER             ********   X   03
IOC$MOVTOUSER             ********   X   03
IOC$PURGDATAP             ********   X   03
IOC$RELDATAP              ********   X   03
IOC$RELMAPREG             ********   X   03
IOC$REQCOM                ********   X   03
IOC$REQDATAP              ********   X   03
IOC$REQMAPREG             ********   X   03
IOC$RETURN                ********   X   03
IOC$WFIKPCH               ********   X   03
IPL$_POWER              = 0000001F
IRP$[_MEDIA             = 00000038
IRP$L_SEGVBN            = 00000048
IRP$W_FUNC              = 00000020
MASKH                   = 00000080
MASKL                   = 08000000
MOVFRUSER                 00000442 R     03
MOVTOUSER                 00000453 R     03
P1                      = 00000000
P2                      = 00000004
P3                      = 00000008
P4                      = 0000000C
P5                      = 00000010
P6                      = 00000014
PR$_IPL                 = 00000012
PRI$_IOCOM              = 00000001
RETURN_STATUS             000003A7 R     03
SCH$QAST                  ********   X   03
SIZ...                  = 00000001
SS$_BADPARAM            = 00000014
SS$_CANCEL              = 00000830
SS$_CTRLERR             = 00000054
SS$_DRVERR              = 0000008C
SS$_NOPRIV              = 00000024
SS$_NORMAL              = 00000001
SS$_OPINCOMPL           = 000002D4
SS$_PARITY              = 000001F4
SS$_TIMEOUT             = 0000022C
UCB$B_DEVCLASS          = 00000040
UCB$B_DEVTYPE           = 00000041
```

```
UCB$B_DIPL              = 0000005E
UCB$B_FIPL              = 0000000B
UCB$K_SIZE              = 000000C2
UCB$L_CRB               = 00000024
UCB$L_DEVCHAR           = 00000038
UCB$L_DEVDEPEND         = 00000044
UCB$L_DPC               = 0000009C
UCB$L_DUETIM            = 0000006C
UCB$L_FPC               = 0000000C
UCB$L_FR3               = 00000010
UCB$L_SVAPTE            = 00000078
UCB$L_XA_ATTN             000000A0
UCB$L_XA_DPR              000000B4
UCB$L_XA_FMPR             000000B8
UCB$L_XA_PMPR             000000BC
UCB$M_ATTNAST           = 00000001
UCB$M_CANCEL            = 00000008
UCB$M_INT               = 00000002
UCB$M_ONLINE            = 00000010
UCB$M_POWER             = 00000020
UCB$M_TIM               = 00000001
UCB$M_TIMOUT            = 00000040
UCB$M_UNEXPT            = 00000002
UCB$V_ATTNAST           = 00000000
UCB$V_CANCEL            = 00000003
UCB$V_INT               = 00000001
UCB$V_POWER             = 00000005
UCB$V_UNEXPT            = 00000001
UCB$W_BCNT              = 0000007E
UCB$W_BOFF              = 0000007C
UCB$W_DEVBUFSIZ         = 00000042
UCB$W_DEVSTS            = 00000068
UCB$W_FUNC              = 0000009A
UCB$W_STS               = 00000064
UCB$W_XA_BAR              000000AE
UCB$W_XA_BARTMP           000000A6
UCB$W_XA_CSR              000000A8
UCB$W_XA_CSRTMP           000000A4
UCB$W_XA_DPRN             000000C0
UCB$W_XA_EIR              000000AA
UCB$W_XA_ERROR            000000B2
UCB$W_XA_IDR              000000AC
UCB$W_XA_WCR              000000B0
VEC$B_DATAPATH          = 00000013
VEC$L_IDB               = 00000008
VEC$L_INITIAL           = 0000000C
VEC$M_PATHLOCK          = 00000080
VEC$S_DATAPATH          = 00000005
VEC$S_MAPREG            = 0000000F
VEC$V_DATAPATH          = 00000000
VEC$V_MAPREG            = 00000000
VEC$W_MAPREG            = 00000010
WORD_MODE                 00000360 R     03
XA$DDT                    00000000 RG    03
XA$K_FNCT2              = 00000004
XA$V_DATAPATH           = 00000000
XA$V_LINK               = 00000001
XA_BAR                    00000002
XA_CANCEL                 00000542 R     03
XA_CONTROL_INIT           00000084 R     03
XA_CSR                    00000004
XA_CSR$M_ATTN           = 00002000
XA_CSR$M_CYCLE          = 00000100
XA_CSR$M_ERROR          = 00008000
XA_CSR$M_FNCT           = 0000000E
XA_CSR$M_GO             = 00000001
XA_CSR$M_IE             = 00000040
XA_CSR$M_MAINT          = 00001000
XA_CSR$V_ATTN           = 0000000D
XA_CSR$V_ERROR          = 0000000F
XA_CSR$V_FNCT           = 00000001
XA_CSR$V_XBA            = 00000004
XA_DEF_BUFSIZ           = 0000FFFF
XA_DEF_TIMEOUT          = 0000000A
XA_DEV_RESET              000005FE R     03
XA_EIR                    00000004
XA_EIR$M_ACLO           = 00000800
XA_EIR$M_DLT            = 00000200
XA_EIR$M_MULTI          = 00001000
XA_EIR$M_NEX            = 00004000
XA_EIR$M_PAR            = 00000400
XA_END                    00000636 R     03
XA_FUNCTABLE              00000038 R     03
XA_IDR                    00000006
XA_INTERRUPT              000004CB R     03
XA_ODR                    00000006
XA_READ_WRITE             0000009E R     03
XA_REGDUMP                000005D9 R     03
XA_REGISTER               000004F5 R     03
XA_RESET_DELAY          = 00000001
XA_SETMODE                000000F8 R     03
XA_START                  0000014B R     03
XA_TIME_OUT               00000472 R     03
XA_TIME_OUTW              0000048A R     03
XA_WCR                    00000000
```

```
                                        +-----------------+
                                        ! Psect synopsis !
                                        +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           000000C2  (  194.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$105_PROLOGUE                 00000064  (  100.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$115_DRIVER                   00000636  ( 1590.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG

                                      +-------------------------+
                                      ! Performance indicators !
                                      +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     35    00:00:00.04    00:00:00.61
Command processing                141    00:00:00.43    00:00:03.91
Pass 1                            551    00:00:16.22    00:00:51.03
Symbol table sort                   0    00:00:02.39    00:00:09.38
Pass 2                            236    00:00:03.61    00:00:15.02
Symbol table output                24    00:00:00.14    00:00:00.24
Psect synopsis output               3    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              992    00:00:22.87    00:01:20.22

The working set limit was 2100 pages.
135207 bytes (265 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2188 non-local and 58 local symbols.
1292 source lines were read in Pass 1, producing 20 object records in Pass 2.
53 pages of virtual memory were used to define 50 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          35
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       12
TOTALS (all libraries)                                  47

2463 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XADRIVER/OBJ=OBJ$:XADRIVER MSRC$:XADRIVER/UPDATE=(ENH$:XADRIVER)+EXECML$/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TUDRIVER
LIS

XADRIVER
LIS

TSDRIVER
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XEDRIVER
LIS

XDDRIVER
LIS